Determining Authorship

Oct 29, 2015
Working with Strings

Check out the documentation...

• The `str` type has lots of great member functions:
  – `find()`
  – `replace()`
  – `strip()`, `lstrip()`, `rstrip()`
  – `join()` — the opposite of `split()`
Working with Strings

Check out the documentation...

• The `str` type has lots of great member functions:
  – `find()`
  – `replace()`
  – `strip()`, `lstrip()`, `rstrip()`
  – `join()` — the opposite of `split()`
find()

• Finds the first position of a word in a text

• Can start looking at some position (inclusive), stop at another position (exclusive)

  – *Optional* arguments!

```python
>>> mobyString.find('me')
5
>>> mobyString.find('me', 7)
20
>>> mobyString.find('me', 22, len(mobyString))
139
```
replace()

• Replaces all occurrences of one string by another
• Can specify the maximum number of substitutions to be made
  – Optional argument!

Try these two:

```python
>>> mobyString.replace('I', 'YOUR-LOYAL-CS931-TEACHER')
>>> mobyString.replace('I', 'YOUR-LOYAL-CS931-TEACHER', 6)
```
strip(), lstrip(), rstrip()

• Removes whitespace at **start and end** of the string
  – lstrip() does that only for the **start** of the string
  – rstrip() does that only for the **end** of the string

• You can specify the string to be stripped as an **optional** argument (defaults to whitespace)
join()

• Joins a list of strings through the specified delimiter (on which the function is called)
  – If the list of words is a single string, the function treats that string as a list of characters (as usual)

Try these two:

```python
>>> delim = ':'
>>> delim.join(['a', 'b', 'c'])
'a:b:c'
>>> delim.join('word')
'w:o:r:d'
```
Determining Authorship

Define Problem

Find Data

Write a set of instructions

Python

Solution

Project Gutenberg
Determining Authorship: Data

Five Books from a Famous Children’s Series

One Book from a Famous Children’s Series
Determining Authorship: Data

Five Books from a Famous Children’s Series

One Book from a Famous Children’s Series

Six Books from Two Famous Children’s Series
Determining Authorship

Define Problem

Find Data

Write a set of instructions

Discern the **Outlier**: The one book that is NOT in the series of the others.

Solution

Python

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences
The Federalist Papers

• 85 articles written in 1787 to promote the ratification of the US Constitution

• In 1944, Douglass Adair guessed authorship
  – Alexander Hamilton (51)
  – James Madison (26)
  – John Jay (5)
  – 3 were a collaboration

• Corroborated in 1964 by a computer analysis

Determining Authorship

Discern the **Outlier**: The one book that is NOT in the series of the others.

1 vs. 2
Stop Words

Stop Words are words that are filtered out in natural language processing.
Stop Words

Stop Words are words that are filtered out in natural language processing.
Stop Words

Stop Words are words that are filtered out in natural language processing

a, able, about, across, after, all, almost, also, am, among, an, and, any, are, as, at, be, because, been, but, by, can, cannot, could, dear, did, do, does, either, else, ever, every, for, from, get, got, had, has, have, he, her, hers, him, his, how, however, i, if, in, into, is, it, its, just, least, let, like, likely, may, me, might, most, must, my, neither, no, nor, not, of, off, often, on, only, or, other, our, own, rather, said, say, says, she, should, since, so, some, than, that, the, their, them, then, there, these, they, this, tis, to, too, twas, us, wants, was, we, were, what, when, where, which, while, who, whom, why, will, with, would, yet, you, your

http://www.textfixer.com/resources/common-english-words.txt
Stop Words

Stop Words are words that are filtered out in natural language processing

a, able, about, across, after, all, almost, also, am, among, an, and, any, are, as, at, be, because, been, but, by, can, cannot, could, dear, did, do, does, either, else, ever, even, every, few, for, from, get, got, had, has, have, he, her, hers, him, his, how, however, if, in, into, is, it, its, just, least, let, like, likely, may, me, might, most, must, my, neither, no, nor, not, of, off, often, on, only, or, other, our, own, rather, said, say, says, she, should, since, so, some, than, that, the, their, their, these, they, this, to, too, twas, us, wants, was, we, were, what, when, where, which, while, who, whom, why, will, with, would, yet, you, your

http://www.textfixer.com/resources/common-english-words.txt
Determining Authorship

Discern the **Outlier**: The one book that is NOT in the series of the others.

1. Calculate the word counts of the stop words in the two books

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>1000</td>
<td>238</td>
<td>483</td>
<td>12</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>102</td>
<td>93</td>
<td>10</td>
<td>0</td>
<td>15</td>
<td>...</td>
</tr>
</tbody>
</table>
Determining Authorship

Discern the **Outlier**: The one book that is NOT in the series of the others.

1. Calculate the word counts of the stop words in the two books
2. Normalize to get word frequencies

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>
Determining Authorship

1. Calculate the word counts of the stop words in the two books
2. Normalize to get word frequencies

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

3. Design a **metric** to compare the two files
   - A metric is a function that defines a **distance** between two things
Determining Authorship

1. Calculate the word counts of the stop words in the two books
2. Normalize to get word frequencies

|     | a   | able  | about | across | after | ...
|-----|-----|-------|-------|--------|-------|-----
| File 1 | .3  | .01   | .003  | .0027  | 0.006 | ...
| File 2 | 0.238 | 0.0932 | 0.0034 | 0.0021 | 0.05  | ...

3. Design a **metric** to compare the two files
   - A metric is a function that defines a **distance** between two things

Write a `compareTwo(list1, list2)` function that returns a **float**.
Determining Authorship

1. Calculate the word counts of the stop words in the two books
2. Normalize to get word frequencies

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>at</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
</tr>
</tbody>
</table>

3. Design a **metric** to compare the two files
   - A metric is a function that defines a **distance** between two things

Write a `compareTwo(list1, list2)` function that returns a `float`
Determining Authorship

Download and extract ACT2-7.zip
Evaluate and run testFiles('output.csv')
Determining Authorship

Download and extract ACT2-7.zip

Evaluate and run testFiles('output.csv')

We are going to modify two things:
  – compareTwo function
  – Write distance matrix to a file
Determining Authorship

Download and extract ACT2-7.zip
Evaluate and run testFiles('output.csv')

We are going to modify two things:

- compareTwo function
- Write distance matrix to a file
Determining Authorship

Download and extract ACT2-7.zip

Evaluate and run testFiles('output.csv')

We are going to modify two things:

- `compareTwo` function
- Write distance matrix to a file
Frequency table in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, 1, 0.003, 0.0027, 0.0006,...],
  [.238, 0.0932, 0.0034, 0.0021, 0.05 ,...],
...
```

frequencies

```
Frequency table in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...],
  [.238, .0932, .0034, .0021, .05 ,...],

frequencies[0]
... 
... 
... 
... ]
```
Frequency table in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...],
  [.238, .0932, .0034, .0021, .005,...],
  frequencies[1]
  ...
  ...
  ...
  ...
]
```
Frequency table in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...],
[.238, 0.932, 0.0034, 0.0021, 0.05 ,...],
```

frequencies[1][0]
Frequency table in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...],
  [.238, .0932, .0034, .0021, .005 ,...],
  ...
  ...]
```

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences
Your task in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...], i
  [.238, .0932, .0034, .0021, .05,...], j
  ...
  ...
  ...
  ...
  ]
```
Your task in `compareTwo()`

We’re representing this:

|       | a   | able | about | across | after | ...
|-------|-----|------|-------|--------|-------|-----
| File 1| .3  | .01  | .003  | .0027  | 0.006 | ...
| File 2| 0.238 | 0.0932 | 0.0034 | 0.0021 | 0.05  | ...

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...], i
  [.238, .0932, .0034, .0021, .05,...], j
  ...
  ...
  ...
  ...
  ]
```
Your task in `compareTwo()`

We’re representing this:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>able</th>
<th>about</th>
<th>across</th>
<th>after</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>.3</td>
<td>.01</td>
<td>.003</td>
<td>.0027</td>
<td>0.006</td>
<td>...</td>
</tr>
<tr>
<td>File 2</td>
<td>0.238</td>
<td>0.0932</td>
<td>0.0034</td>
<td>0.0021</td>
<td>0.05</td>
<td>...</td>
</tr>
</tbody>
</table>

As a list of lists!

```
[ [.3, .1, .003, .0027, .0006,...], i
  [.238, .0932, .0034, .0021, .05 ,...], j
  ...
  ...
  ...
  ...
 ]
```
Determining Authorship

Download and extract `ACT2-7.zip`
Evaluate and run `testFiles('output.csv')`

We are going to modify two things:

- `compareTwo` function
- Write distance matrix to a file
One way of doing it

\[ \text{len of frequencies}[i] = \text{len of any other frequencies}[j] = \text{len of word list} \]

val = 0.0

for word in range(0, len(frequencies[i])):
    freqsI = frequencies[i]
    freqsJ = frequencies[j]
    val = val + abs(freqsI[word] - freqsJ[word])

return val
“How a Computer Program Helped Reveal J. K. Rowling as Author of A Cuckoo’s Calling”

Run testFiles('output.csv')

This matrix looks kind of familiar...
Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

```python
myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file\n')
myFile.write(str(myNum))
myFile.write('\n')
myFile.close()
```
Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file
')
myFile.write(str(myNum))
myFile.write('\n')
myFile.close()
Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

```python
myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file\n')
myFile.write(str(myNum))
myFile.write('\\n')
myFile.close()
```
# Generating the file

For each file, create a string row with all the values in the corresponding list row in distMatrix, with commas in between.

```python
for i in range(0, len(FILE_LIST)):
    row = '' + FILE_LIST[i]

    # Loop through the columns in the current list row
    for val in distMatrix[i]:
        row = row + ',' + str(val)

    # At this point, we created our string row.
    # We want to write this row into our csv
    outFile.write(row)

    # Need a newline at the end of each string row
    outFile.write('
')

# Finalize the new file by closing it
outFile.close()
```
Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

Open the CSV file in Google Spreadsheets. Use conditional formatting to look for patterns.
What’s Your Answer?

Discern the **Outlier:**
The one book that is NOT in the series of the others.

<table>
<thead>
<tr>
<th>File</th>
<th>Title</th>
<th>Series</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file2.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file3.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file4.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file5.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file6.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s Your Answer?

Discern the **Outlier**: The one book that is NOT in the series of the others.

<table>
<thead>
<tr>
<th>File</th>
<th>Title</th>
<th>Series</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.txt</td>
<td>Wonder Wizard of Oz</td>
<td>Oz</td>
<td></td>
</tr>
<tr>
<td>file2.txt</td>
<td>Alice’s Adventures in Wonderland</td>
<td>Alice in Wonderland</td>
<td></td>
</tr>
<tr>
<td>file3.txt</td>
<td>Dorothy and the Wizard in Oz</td>
<td>Oz</td>
<td></td>
</tr>
<tr>
<td>file4.txt</td>
<td>Emerald City of Oz</td>
<td>Oz</td>
<td></td>
</tr>
<tr>
<td>file5.txt</td>
<td>Royal Book of Oz</td>
<td>Oz</td>
<td></td>
</tr>
<tr>
<td>file6.txt</td>
<td>Glinda of Oz</td>
<td>Oz</td>
<td></td>
</tr>
</tbody>
</table>
The Wizard of OZ

- About 40 Books, written by 7 different authors

Lyman Frank Baum (1856-1919)

Ruth Plumly Thompson

Published in 1921

http://www.ssc.wisc.edu/~zzeng/soc357/OZ.pdf
What’s Your Answer?

Discern the Outlier:
The one book that is NOT in the series of the others.

<table>
<thead>
<tr>
<th>File</th>
<th>Title</th>
<th>Series</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.txt</td>
<td>Wonder Wizard of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file2.txt</td>
<td>Alice’s Adventures in Wonderland</td>
<td>Alice in Wonderland</td>
<td>Lewis Carroll</td>
</tr>
<tr>
<td>file3.txt</td>
<td>Dorothy and the Wizard in Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file4.txt</td>
<td>Emerald City of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file5.txt</td>
<td>Royal Book of Oz</td>
<td>Oz</td>
<td>Ruth Plumly Thompson</td>
</tr>
<tr>
<td>file6.txt</td>
<td>Glinda of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
</tbody>
</table>
## What’s Your Answer?

<table>
<thead>
<tr>
<th>File</th>
<th>Title</th>
<th>Series</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.txt</td>
<td>Wonder Wizard of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file2.txt</td>
<td>Alice’s Adventures in Wonderland</td>
<td>Alice in Wonderland</td>
<td>Lewis Carroll</td>
</tr>
<tr>
<td>file3.txt</td>
<td>Dorothy and the Wizard in Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file4.txt</td>
<td>Emerald City of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
<tr>
<td>file5.txt</td>
<td>Royal Book of Oz</td>
<td>Oz</td>
<td>Ruth Plumly Thompson</td>
</tr>
<tr>
<td>file6.txt</td>
<td>Glinda of Oz</td>
<td>Oz</td>
<td>Lyman Frank Baum</td>
</tr>
</tbody>
</table>
Stuff to do

• Think about Project 2 ideas for your proposal
  – Initial proposal due tomorrow
  – Revised proposal due on Nov 6

• Revisit the activities we did to get more practice with Python
Tools you’ve learned

• Reading and writing files (ACT 2-2, 2-7)
• String processing: split(), find(), etc. (ACT 2-2, 2-6)
• Lists and dictionaries (ACT 2-5)
• Iterating over data: two approaches to for-loops (HW 2-4)
• Summaries statistics like counts, averages, min/max (ACT 2-3)